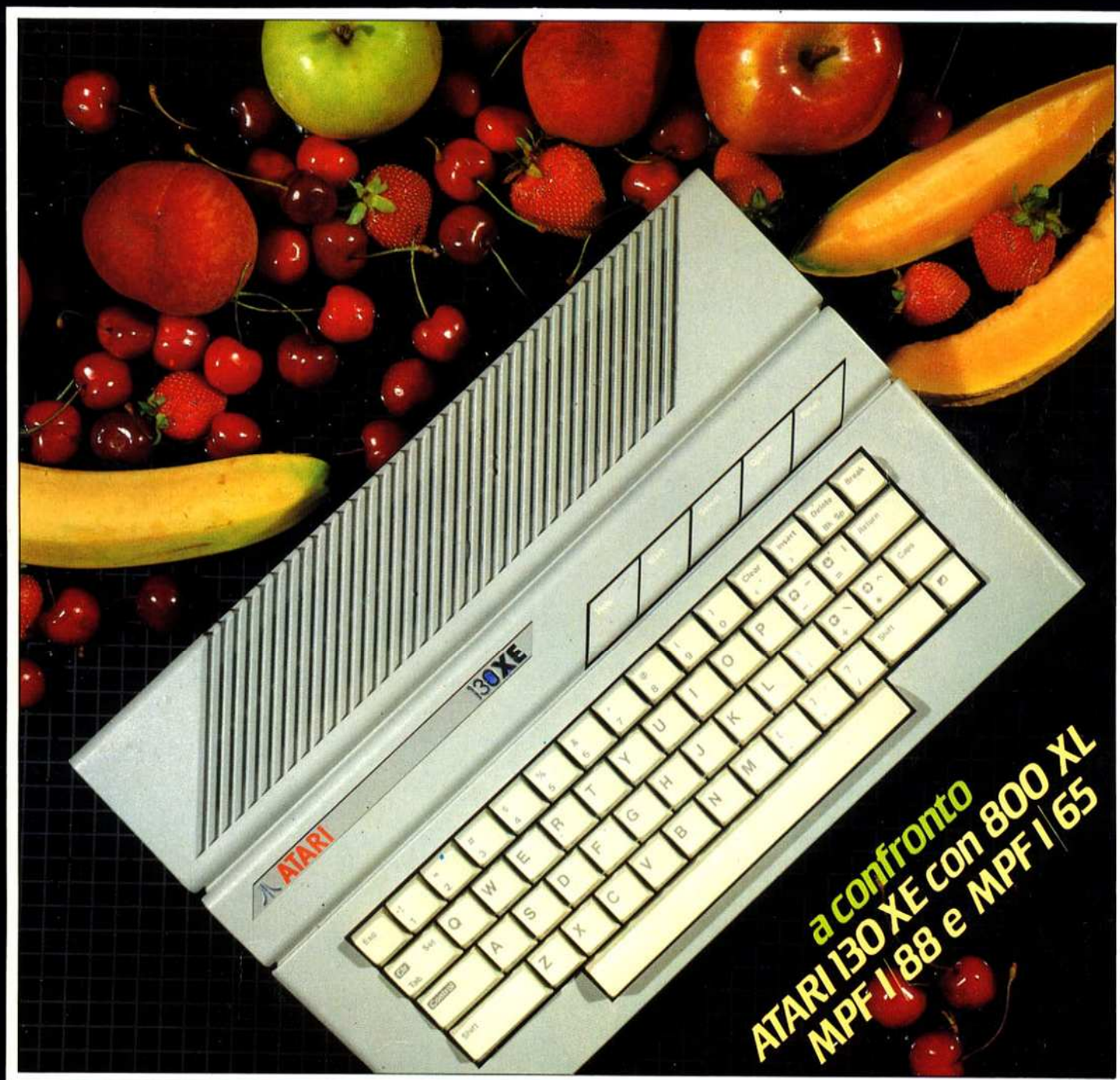
MICRO & PERSONAL computer

55 56

lire 4.000

m&p COMPUTER - Lug.-Ago. 1985 - n. 55-56 - Anno VI - mensile - Sped. abb. post. gr. III 70%

a confronto
ATARI 130 XE con 800 XL
MPF I/88 e MPF I/65

SOFTWARE: EASYFAMILY
GRAFICA: I PROGRAMMI PER SINCLAIR
RUBRICHE: APPLE. COMMODORE. MSX. SHARP. TEXAS. MACINTOSH
MERCATO: DOVECOME COMPUTER

SOMMARIO

55-56

28 40 35 99 18

dove come computer

Copertina - Fotografia: Alessandro Neri **Grafica:** Diana Santosuosso

di PAOLO BAGNARESI

TI 99/4A

XEDITOR: EDITOR PER XASSEMBLE

Pubblichiamo questo mese il promesso XEDITOR, che è l'EDITOR per comporre SORGENTI adatte ad essere assemblate con XASSEMBLE. Infatti, XASSEMBLE (assemblatore per EXTENDED BASIC con espansione di memoria 32 K, sia laterale, tipo ESSEMME-CI, sia contenuta nel PERIPHERAL BOX), pubblicato nei mesi di Marzo '85 e Maggio '85, permette l'assemblaggio di SORGENTI da file, cosa che con la MINI-MEMORY non è possibile fare, o anche l'assemblaggio di SORGENTI battute direttamente da tastiera, come la MINI-MEMORY. È importante notare come sia preferibile comporre una SORGENTE con un EDITOR e poi farla assemblare ad un assemblatore (caso 1), piuttosto che immettere una dopo l'altra le istruzioni da tastiera (caso 2). Nel primo caso la SORGENTE non va persa durante l'assemblaggio; potremo così variarla successivamente, correggerla, adattarla a nuove e diverse esigenze con il minimo di fatica. Nel secondo caso la SORGENTE va invece persa durante l'assemblaggio: ogni successiva correzione non è più possibile e, se il programma così composto si dimostra inadatto durante il «RUN» (leggi «CALL LINK...»), occorre ribatterlo per intero. Si consiglia caldamente, pertanto, di usare l'XEDITOR proposto questo mese, non appena la sorgente supera le 10-20 istruzioni.

XEDITOR è in linguaggio BASIC (versione EXTENDED).
È consigliabile il possesso dell'espansione di memoria, anche se il programma può funzionare anche senza espansione. In quest'ultimo caso, tuttavia, i files generabili sono veramente limitati. In ogni modo, una CALL FILES (1) permette la creazione di files più lunghi.

CARATTERISTICHE DI XEDITOR

1) Schermo a 32 colonne, colore celeste, caratteri in bianco, ogni linea testo può essere al massimo di 28 caratteri.
2) Cursore al centro, sulla 12esima riga. Scroll normale di una riga con ENTER. Possibilità di scroll dello schermo di 23 righe in blocco, sia in alto, che in basso.
3) Ulteriore possibilità di scroll in alto e in basso del numero di righe desiderate.
4) Possibilità di evidenziare il numero di riga a fianco di ciascuna riga-testo. Segnalazione perenne del numero di riga attualmente sotto il cursore e dell'ultimo numero di riga del record.
5) Opzione di spostamento di una o più righe all'interno del file.
6) Possibilità di cambiare automaticamente un determinato segmento di stringa di testo su tutto il file, con un'altro segmento selezionabile da parte dell'utente, e ciò per tutte le volte che il primo segmento è presente nel testo. È la «Replace String» dell'EDITOR/ASSEMBLER.
7) Inserimento facilitato di nuove righe-testo (Insert).
8) Cancellazione di una o più righe a piacere (Erase).
9) OLD e SAVE sia da cassetta che da disco.
10) Possibilità di MERGE TOTALE di un file esterno nel file attualmente in memoria (come l'EDITOR/ASSEMBLER), e, anche, di un MERGE PARZIALE, cosa che neppure l'EDITOR/ASSEMBLER Texas è in grado di fare. È una funzione che si ritrova solo nel TI-WRITER Texas. Queste due funzioni sono usabili sia con disco che con cassetta.
11) Opzione di OLD e SAVE PARZIALE, sia per disco che per cassetta. Anche questa opzione è presente solo nel TI-WRITER.
12) Stampa eventuale del file, con possibilità di stampa anche dei numeri di linea, cosa che neppure il TI-WRITER è in grado di fare.

Si può facilmente intuire, da quanto elencato, che XEDITOR è veramente molto versatile: ha delle funzioni (MERGE PARZIALE, SAVE e OLD PARZIALE), che neppure l'EDITOR/ASSEMBLER Texas possiede, e che, detto per inciso, sono molto comode.

```
100 !************************
110 !*     XEDITOR
120 !*PAOLO BAGNARESI 5.1985
130 !* Tel. (02)-514.202
140 !************************

150 ! Inizializzazione

160 OPTION BASE 0 :: MX=629
:: DIM Q$(630):: DE$="C" ::
FE$="V" :: ON ERROR 1290

170 SHLI=0 :: C$="~LC=" :: V
AL$="UDNLRMISOEF" :: ST$="RS
232.BA=4800.DA=8" :: SL$="N"
 :: CALL CLEAR :: RIG1=20 ::
 RIG2=23

180 L$=RPT$(CHR$(131),28)::
I$=RPT$(CHR$(132),28):: CALL
 CHAR(129,"8080808080808080"
,130,"0101010101010101",131,
"000000FF",132,"FF",133,"000
0000000000FF")

190 CALL CHAR(140,"004E50504
E407000",141,"5054545454547404
07"):: C$=CHR$(140):: UL$=CH
R$(141)

200 K$=RPT$(CHR$(133),28)::
CALL SCREEN(6):: FOR T=0 TO
14 :: CALL COLOR(T,16,6):: N
EXT T

210 ! Display SCHERMO DI ENT
RATA

220 DISPLAY AT(1,1):I$: :TAB
(11);"XEDITOR": :L$:" 1985 P
aolo Bagnaresi & M&P":L$:"Ge
nera files assemblabili":"co
n XASSEMBLER. Il file"

230 DISPLAY AT(10,1):"genera
to e' stampabile.":L$:"FCTN
X  abilita gli specials":L$:
"FCTN E da' scroll in su'":L
$:"ENTER accetta la linea":L
$

240 DISPLAY AT(18,2):C$;"=Li
nea sotto Cursore":L$:" ";UL
$;"=Ultima Linea file":L$

250 FOR T=1 TO 24 :: CALL VC
HAR(T,2,130,1):: CALL VCHAR(
T,31,129,1):: NEXT T :: GOSU
B 1260

260 DISPLAY AT(18,1):"File N
uovo o Vecchio? (N/V)";FE$:L
$ :: ACCEPT AT(18,28)VALIDAT
E("NV")SIZE(-1)BEEP:FE$ :: I
F FE$="V" THEN 1030

270 CALL CLEAR :: RIG1=1 ::
RIG2=3

280 ! ROUTINE PRINCIPALE EDI
TOR

290 FOR T=24+(I>MX-12)*(I-MX
+12)TO 12-R STEP -1 :: DISPL
AY AT(T,1):Q$(I+T-12):: NEXT
 T :: DISPLAY AT(1,24):C$;I+
1 :: DISPLAY AT(2,24):UL$;TO
P+1 :: IF SHLI<0 THEN GOSUB
1230

300 IF I>MX-12 THEN CALL HCH
AR(13+MX-I,1,32,32*(I-MX+12)
)

310 ACCEPT AT(12,1)SIZE(-28)
:Q$(I):: CALL KEY(0,K,S):: I
=I+1 :: IF I>MX THEN I=MX

320 TOP=MAX(TOP,I):: R=(I<11
```

```
)*-I+(I>10)*-11 :: IF K=10 T
HEN 340 ELSE IF K=11 THEN 33
0 ELSE 290

330 GOTO 1200

340 DISPLAY AT(1,1)BEEP:"U(p
 or D(own, N(line, L(ine":"R
(eplace, M(ove, I(nsert":"S(
ave, O(ld, E(rase,F(ine ";M$
:L$

350 RIG1=1 :: RIG2=3 :: ACCE
PT AT(3,27)VALIDATE(VAL$)SIZ
E(-1)BEEP:M$ :: IF M$<>"L" T
HEN 380

360 DISPLAY AT(1,1):"
 Linee scroll":"Quante in gi
u'? (Ex. 7)":"Quante in su'?
 (Ex.-7): ";O$ :: GOSUB 1240

370 ACCEPT AT(3,25)SIZE(-4)V
ALIDATE(DIGIT,"+-")BEEP:O$ :
: IF O$="" THEN 1210 ELSE SC
=VAL(O$):: GOTO 510

380 IF M$="" THEN 1210 ELSE
ON POS(VAL$,M$,1)+1 GOTO 290
,480,500,530,510,560,670,400
,780,1020,440,1320

390 ! INSERISCE LINEE

400 DISPLAY AT(1,1):"Inser.
linee da lin. cursore":"Quan
te linee? : ";SERT$: : :: GO
SUB 1240 :: ACCEPT AT(2,17)V
ALIDATE(DIGIT)SIZE(-3)BEEP:S
ERT$

410 IF SERT$="" THEN 340 ELS
E SERT=VAL(SERT$)

420 FOR W=MAX(TOP,I)TO I-1 S
TEP -1 :: Q$(W+SERT)=Q$(W)::
 NEXT W :: FOR T=0 TO SERT-1
 :: Q$(I-1+T)="" :: NEXT T :
: TOP=TOP+SERT :: GOTO 1210

430 ! CANCELLA UNA O PIU' LI
NEE

440 DISPLAY AT(1,1):"Erase l
inee da linea cursore":"Una
 o piu' linee ? : ";NLI$: :
:: GOSUB 1240

450 ACCEPT AT(2,23)SIZE(-3)V
ALIDATE(DIGIT)BEEP:NLI$ :: I
F NLI$="" THEN 1210 ELSE NL=
VAL(NLI$):: IF NL+I-1>TOP TH
EN NL=TOP-I+1

460 NL=NL*-(NL>0):: FOR W=I-
1+NL TO MAX(TOP,I):: Q$(W-NL
)=Q$(W):: NEXT W :: FOR T=W-
1-NL TO W-1 :: Q$(T)="" :: N
EXT T :: TOP=TOP-NL :: GOTO
1200

470 ! MOSTRA LO SCHERMO A LI
NEE PIU' BASSE

480 I=I-22 :: GOTO 1200

490 ! MOSTRA LO SCHERMO A LI
NEE PIU' ALTE

500 SC=23

510 I=I+SC+1 :: GOSUB 1220 :
: GOTO 1200

520 ! MOSTRA NUMERI DI LINEA

530 SHLI=SHLI=0 :: IF SHLI<>
0 THEN 1210

540 FOR T=24 TO 12-R STEP -1
 :: CALL VCHAR(T,2,32,1):: N
EXT T :: GOTO 1210

550 ! REPLACE STRING

560 DISPLAY AT(1,1):"Replace
 String: /Old/New/ ":L$:RP$:
L$: : :: ACCEPT AT(3,1)SIZE(
-28)BEEP:RP$ :: IF RP$="" TH
EN 340 ELSE CALL HCHAR(5,1,3
2,640):: ANS=0

570 P1=POS(RP$,"/",1):: IF P
1=0 THEN 560 ELSE P2=POS(RP$
,"/",P1+1):: IF P2=0 THEN 56
0 ELSE P3=POS(RP$,"/",P2+1):
: IF P3=0 THEN 560

580 RP1$=SEG$(RP$,P1+1,P2-P1
-1):: RP2$=SEG$(RP$,P2+1,P3-
P2-1):: I=SAL

590 FOR I=0 TO TOP :: P4=POS
(Q$(I),RP1$,1):: IF P4=0 THE
N 640 ELSE R=(I<11)*-I+(I>10
)*-11

600 FOR T=24 TO 12-R STEP -1
 :: DISPLAY AT(T,1):Q$(I+T-1
2):: NEXT T :: DISPLAY AT(1,
21):C$;I+1 :: CALL HCHAR(12,
P4+1,30)

610 IF ANS=1 THEN 630 ELSE D
ISPLAY AT(1,1):L$:RP$:"Repla
ce? [S/N/A(ll/B(asta] ";AN$:
L$

620 ACCEPT AT(3,28)SIZE(-1)V
ALIDATE("SNAB")BEEP:AN$ :: I
F AN$="N" THEN 640 ELSE IF A
N$="B" THEN 650 ELSE IF AN$=
"A" THEN ANS=1

630 Q$(I)=SEG$(Q$(I),1,P4-1)
&RP2$&SEG$(Q$(I),P4+P2-P1-1,
LEN(Q$(I))):: Q$(I)=SEG$(Q$(
I),1,28)

640 NEXT I

650 I=SAL+1 :: GOTO 1210

660 ! MUOVE LINEE

670 DISPLAY AT(1,1):"Da Riga
 a Riga e";CHR$(30);C$;I;UL$
;TOP+1:"mette dopo Riga :";M
NR$:"Ex 4,6,23 Riga 4-5-6 do
po 23" :: IF SHLI=0 THEN I=I
-1 :: GOSUB 1230 :: I=I+1

680 GOSUB 1240 :: ER=-2 :: A
CCEPT AT(2,18)BEEP SIZE(-11)
VALIDATE(DIGIT,","):MNR$ ::
IF MNR$="" THEN 760 ELSE CAL
L S(MNR$,T,MN1,MN2,VE$,ER)

690 IF ER>0 THEN 680 ELSE MN
3=VAL(VE$)-1

700 TOP=MAX(TOP,I):: IF MN1<
MN3 AND MN3<MN2 OR MN1>TOP+1
 OR MN2>TOP+1 OR MN3>TOP+1 O
R MN1<0 OR MN2<0 OR MN3<0 TH
EN 680

710 SP=MN2-MN1+1 :: FOR T=0
TO SP-1 :: Q$(TOP+T+1)=Q$(MN
1+T):: NEXT T :: IF MN3<MN1
THEN 740

720 FOR T=0 TO MN3-MN2 :: Q$
(MN1+T)=Q$(MN2+T+1):: NEXT T

730 FOR T=0 TO SP-1 :: Q$(MN
3-SP+T+1)=Q$(TOP+T+1):: NEXT
 T :: FOR T=TOP+1 TO TOP+SP
:: Q$(T)="" :: NEXT T :: GOT
O 760

740 FOR T=0 TO MN1-MN3-1 ::
Q$(MN2-T)=Q$(MN1-T-1):: NEXT
 T

750 FOR T=0 TO SP-1 :: Q$(MN
3+T+1)=Q$(TOP+T+1):: NEXT T
:: FOR T=TOP+1 TO TOP+SP ::
Q$(T)="" :: NEXT T

760 IF SHLI<>0 THEN 1200 ELS
E 540

770 ! SAVE ROUTINE : STAMPA,
 DISCO O CASSETTA

780 DISPLAY AT(1,1):L$:"Regi
stra su Disco, Cassetta,":"o
```

ERRATA CORRIGE E PRECISAZIONI

1) A pag. 119 del numero 51 di m&p (marzo '85), al punto 2), è sottinteso che si debba fare una SAVE dopo la CALL LINK («RILOC»). Il programma così salvato sarà fatto oggetto di OLD al punto 4), «ORA ESEGUITE UNA OLD DI LINEA 25000». La cosa non è stata ovvia per molti lettori, e forse proprio non lo è! Mille scuse per la svista.

2) A pag. 120 dello stesso numero, l'indirizzo ESAdecimale che porta alla BLOAD non è > FD8C, come segnalato, ma > FD90. Tuttavia, l'indirizzo decimale fornito è esatto.

3) Nello stesso numero 51, tutte le volte che si cita SPOLINEXT, si intende il programma SPOLIN in formato CALL LOAD per EXT. BASIC.

4) A pag. 86 del numero di 53 (maggio '85), al punto 2) RESTOR MOVB @LOAD, CS1PAB si deve leggere: RESTOR MOVB @LOAD, @CS1PAB. È stato omesso il simbolo @ prima di CS1PAB.

5) A pag. 87, nello stesso numero 53, al punto 7) SINTASSI, non tenere conto degli esempi riportati. Essi sono stati alterati in sede di impaginazione. Il concetto semplice che si voleva esprimere è che non ci deve essere più di uno spazio né tra LABEL e ISTRUZIONE, né tra ISTRUZIONE e OPERANDO.

6) Per una svista tipografica, il listato XASSEMBLE apparso a pag. 91 dello stesso numero di Maggio è mancante di 7 caratteri, relativi alla linea 380. Di nuovo scuse. Questa è la linea 380 completa:

380 IF IN$< > " " THEN IF SEG$(IN$, LEN (IN$), 1) = " " THEN IN$ = SEG$ (IN$, 1, LEN (IN$) – 1):: GO TO 380

7) Se assemblate un secondo file con XASSEMBLE, sappiate che, se avete risposto N alla domanda REF? nel primo file, non avrete più disponibili le REF nel secondo file. Se le REF sono proprio necessarie, occorre uscire dal programma e poi rientrare. Salvo particolari ragioni (vedi sotto al punto 8), evitare di rispondere N alla domanda REF.

8) Assemblando un file con XASSEMBLE, ed utilizzando l'opzione «COPY», verrà di nuovo chiesto REF? (S/N). Occorre rispondere N (MAIUSCOLO), altrimenti tutti i LABEL definiti nel file precedente vengono perduti. Chi volesse eliminare il rischio di una dimenticanza, modifichi la linea 300 inserendo all'inizio: IF IN$< > «COPY» THEN...

```
 Stampa (D/C/S) ";DE$ :: ACC
EPT AT(3,18)VALIDATE("DCS")S
IZE(-1)BEEP:DE$

790 IF DE$="" THEN 340 ELSE
DISPLAY AT(1,1):C$;I;UL$;TOP
+1;"Save, Device? (";DE$;")"
:L$

800 ON POS(" DCS",DE$,1)GOTO
 780,850,850,930

810 ! Save su cassetta

820 OPEN #2:"CS1",FIXED 192,
OUTPUT :: G$=""

830 W$=STR$(LEN(Q$(Y)))&" "&
Q$(Y):: IF LEN(G$)+LEN(W$)<1
93 THEN G$=G$&W$ :: Y=Y+1 EL
SE PRINT #2:G$ :: G$=""

840 IF Q$(Y-1)=" END" OR Q$(
Y-1)=" COPY" OR Y>Z THEN PRI
NT #2:G$ :: CLOSE #2 :: GOTO
 290 ELSE 830

850 DISPLAY AT(3,1):TOT$:L$
:: ACCEPT AT(3,1)SIZE(-28)BE
EP:TOT$ :: IF TOT$="" THEN F
SL$="N" :: GOTO 340 ELSE CAL
L S(TOT$,RT1,RT2,RT3,DR$,ER)
:: IF ER>0 THEN 850 ELSE IF
DR$="" THEN FSL$="N" :: GOTO
 340

860 IF RT1>-1 THEN 850 ELSE
IF RT2>-1 THEN Y=RT2 :: Z=MI
N(RT3,TOP)ELSE Y=0 :: Z=MAX(
TOP,I)

870 IF TOP=1 AND Q$(0)="" AN
D Q$(1)="" THEN 340 ELSE IF
DE$="C" THEN 820 ELSE OPEN #
1:DR$,OUTPUT

880 FOR T=Y TO Z :: IF FSL$=
"S" THEN PRINT #1:T+1;TAB(7)
;Q$(T)ELSE PRINT #1:Q$(T)

890 IF Q$(T)=" END" OR Q$(T)
=" COPY" THEN 910

900 NEXT T

910 CLOSE #1 :: FSL$="N" ::
IF DE$="S" THEN ST$=TOT$ ::
TOT$=TEMP$ :: GOTO 270 ELSE
TEMP$=TOT$ :: GOTO 270

920 ! ROUTINE DI STAMPA

930 DISPLAY AT(3,1):"Stampa
Numeri Linee ?(S/N) ";SL$ ::
 ACCEPT AT(3,28)VALIDATE("SN
")SIZE(-1)BEEP:SL$ :: IF SL$
="S" THEN FSL$="S"

940 DISPLAY AT(1,1):"OPEN:"
:: TOT$=ST$ :: GOTO 850

950 ! Old Cassette

960 OPEN #1:"CS1",INPUT ,FIX
ED 192

970 LINPUT #1:G$

980 LU$=SEG$(G$,1,POS(G$," "
,1)-1):: LULU=LEN(LU$):: LU=
VAL(LU$):: Q$(Z)=SEG$(G$,LUL
U+2,LU):: Z=Z+1 :: G$=SEG$(G
$,LULU+2+LU,LEN(G$))

990 IF SEG$(Q$(Z-1),1,4)=" E
ND" OR SEG$(Q$(Z-1),1,5)=" C
OPY" THEN Z=Z-1 :: GOTO 1120
 ELSE IF RT2>-1 THEN Y=Y+1 :
: IF Y<RT2+1 THEN Z=Z-1 ELSE
 IF Y>RT3 THEN Z=Z-1 :: GOTO
 1120

1000 IF SEG$(G$,1,1)=" " OR
SEG$(G$,1,1)="" THEN 970 ELS
E 980

1010 ! OLD ROUTINE

1020 DISPLAY AT(1,8):"Load F
ile"

1030 DISPLAY AT(RIG1,1):"Old
 Disk o Cass.? (D/C) ";DE$:L
$:L$ :: ACCEPT AT(RIG1,25)VA
LIDATE("DC")SIZE(-1)BEEP:DE$
 :: IF DE$="" THEN 340

1040 IF DE$="" THEN 340 ELSE
 IF RIG1<20 THEN DISPLAY AT(
RIG1-1,1)SIZE(17):C$;I;UL$;T
OP+1

1050 DISPLAY AT(RIG2,1):TOT$
:K$ :: ACCEPT AT(RIG2,1)SIZE
(-28)BEEP:TOT$ :: IF TOT$=""
 THEN 340 ELSE CALL S(TOT$,R
```

```
T1,RT2,RT3,DR$,ER):: IF ER T
HEN 1050

1060 Y=0 :: FLG=0 :: IF RT1>
-1 AND RT3-RT2+TOP>MX THEN D
ISPLAY AT(20,1):L$:"Troppe r
ighe : max";MX;", ma":"per M
erge Righe max =";MX-TOP:L$
:: GOTO 1050

1070 IF RT1=-1 THEN Z=0 :: L
W=MX :: GOTO 1090 ELSE Z=RT1
+1 :: IF RT1>-1 AND RT2>-1 T
HEN NTOP=TOP+RT3+1-RT2 ELSE
RT2=0 :: NTOP=MX :: RT3=NTOP
-1+RT2-TOP

1080 LW=RT1+1+RT3+1-RT2 :: F
OR T=0 TO TOP-RT1-1 :: Q$(NT
OP-T)=Q$(TOP-T):: NEXT T

1090 TEMP$=TOT$ :: IF DE$="C
" THEN 960 ELSE OPEN #1:DR$,
INPUT

1100 ! Old da disco

1110 LINPUT #1:Q$(Z):: Q$(Z)
=SEG$(Q$(Z),1,28):: IF EOF(1
)THEN 1120 ELSE Z=Z+1 :: IF
RT2<0 THEN 1110 ELSE Y=Y+1 :
: IF Y<RT2+1 THEN Z=Z-1 :: G
OTO 1110 ELSE IF Y>RT3 THEN
Z=Z-1 ELSE IF Z>LW-1 THEN FL
G=1 ELSE 1110

1120 IF RT1>-1 THEN 1160

1130 FOR T=1 TO TOP-Z :: Q$(
Z+T)="" :: NEXT T :: TOP=Z

1140 CLOSE #1 :: I=13*-(RT1=
-1)*-(TOP>13)+RT1+2*-(RT1>-1
):: CALL CLEAR :: RIG1=1 ::
RIG2=3 :: GOTO 1200

1150 ! Risistema dopo MERGE

1160 OFS=RT3+1-RT2-(Z-RT1)::
 IF OFS<1 THEN TOP=TOP+Z-RT1
 :: GOTO 1140

1170 FOR T=0 TO TOP-RT1-1 ::
 Q$(Z+T+1)=Q$(LW+T):: NEXT T
 :: TOP=TOP+Z-RT1 ! Ricompat
ta le linee

1180 FOR T=1 TO OFS :: Q$(TO
P+T)="" :: NEXT T :: GOTO 11
40 ! Elimina elementi superi
ori a TOP

1190 ! ROUTINE LIMITI DISPLA
Y AT

1200 I=-(I>0)*(I-1)

1210 I=-(I>0)*(I-1):: R=(I<1
1)*-I+(I>10)*-11 :: TOP=MAX(
TOP,I+1):: CALL HCHAR(1,1,32
,384-(R+1)*32):: IF FLG=0 TH
EN 290 ELSE FLG=0 :: GOSUB 1
290 :: GOTO 290

1220 I=(TOP+1)*-(I>TOP)+I*-(
I<TOP):: RETURN

1230 FOR T=24 TO 12-R STEP -
1 :: CALL VCHAR(T,2,ASC(SEG$
(STR$(I+T-12+1),LEN(STR$(I+T
-12+1)),1)),1):: NEXT T :: R
ETURN

1240 CALL HCHAR(12,2,30):: R
ETURN

1250 ! Premi un tasto

1260 DISPLAY AT(22,1)BEEP:L$
:TAB(5);"_ Premi un tasto _"
:K$

1270 CALL KEY(0,K,S):: IF S=
0 THEN 1270 ELSE RETURN

1280 ! Routine errori

1290 ON ERROR 1300 :: RIG1=1
 :: RIG2=3 :: CLOSE #1

1300 ON ERROR 1290 :: DISPLA
Y AT(18,1):L$:"ERRORE: OPEN,
 MEMORY FULL O":TAB(10);"ALT
RO":L$ :: GOSUB 1260 :: IF R
IG1=1 THEN RETURN 340 ELSE R
ETURN 170

1310 ! Routine di FINE

1320 DISPLAY AT(1,1):"vuoi l
a Fine, Nuovo testo, ":"o Co
ntinuare? (F/N/C): ";FIN$: :
 :: ACCEPT AT(2,24)SIZE(-1)V
ALIDATE("FNC")BEEP:FIN$

1330 IF FIN$="F" THEN 1480 E
LSE IF FIN$="C" THEN 1350

1340 FOR T=0 TO TOP :: Q$(T)
="" :: NEXT T :: TOP,I,R=0

1350 GOTO 1210

1360 ! Subprogramma smistame
nto istruzioni OLD, SAVE, MU
OVE LINEE

1370 SUB S(X$,R1,R2,R3,DR$,E
R):: ER=0 :: ON ERROR 1400 :
: IF X$="" THEN 1400

1380 P1,P2,P3=0 :: R1,R2,R3=
-1 :: FOR T=1 TO LEN(X$):: I
F SEG$(X$,T,1)<>CHR$(32)THEN
 X$=SEG$(X$,T,LEN(X$)):: GOT
O 1410

1390 NEXT T :: GOTO 1410

1400 ER=1 :: GOTO 1470

1410 P1=POS(X$,",",1):: IF P
1=0 THEN 1460 ELSE P2=POS(X$
,",",P1+1):: IF P2>0 THEN P3
=POS(X$,",",P2+1)

1420 IF P1>0 AND P2=0 AND P3
=0 THEN R1=VAL(SEG$(X$,1,P1-
1))-1 :: GOTO 1460

1430 IF P1>0 AND P2>0 AND P3
>0 THEN R1=VAL(SEG$(X$,1,P1-
1))-1 ELSE 1450

1440 R2=VAL(SEG$(X$,P1+1,P2-
P1-1))-1 :: R3=VAL(SEG$(X$,P
2+1,P3-P2-1))-1 :: IF R3<R2
OR R1<0 OR R2<0 THEN 1400 EL
SE 1460

1450 IF P1>0 AND P2>0 AND P3
=0 THEN R2=VAL(SEG$(X$,1,P1-
1))-1 :: R3=VAL(SEG$(X$,P1+1
,P2-P1-1))-1 :: IF R3<R2 OR
R2<0 THEN 1400 ELSE 1460 ELS
E 1400

1460 P4=MAX(P1,P2):: P4=MAX(
P4,P3):: IF P4=0 THEN DR$=X$
 ELSE DR$=SEG$(X$,P4+1,LEN(X
$))

1470 ON ERROR 1290 :: SUBEND

1480 END
```

Per contro, manca in XEDITOR la possibilità di trattare opportunamente le 80 colonne in sede di stampa. XEDITOR non è inteso come normale WORD-PROCESSOR, ma come EDITOR dedicato a files da assemblare.

CARATTERISTICHE DEL FILE GENERATO

Il file generato da XEDITOR è DISPLAY/VARIABLE 80, se su disco, è DISPLAY/FIXED 192 se su cassetta. Il formato DIS/VAR 80 per disco è il formato normale per TI-WRITER e per EDITOR/ASSEMBLER Texas. I files su disco generati da XEDITOR sono quindi scambiabili con quelli generati dai due suddetti moduli: notare, però, che XEDITOR tronca ogni record al ventottesimo carattere. Il file su

cassetta è invece compatibile solo con XASSEMBLE. Infatti in questo file, DISPLAY FIXED 192, il testo viene compattato in maniera molto particolare, per risparmiare tempo durante il SAVE (OUTPUT) e la OLD (INPUT). Ogni record del file al momento di una SAVE CS1. contiene:
1) Lunghezza, sotto forma di stringa, della stringa che segue.
2) Un byte vuoto (codice ASCII 32).
3) La stringa effettiva, la cui lunghezza è indicata al punto 1.
4) Lunghezza della seconda stringa.
5) Altro byte vuoto.
6) Seconda stringa.
Si continua così fino a che lo spazio residuo nel record non è sufficiente a contenere la triade
LUNGHEZZA + SPAZIO + STRINGA.
A questo punto le nuove stringhe vengono immesse nel record FIXED 192 successivo. Vedi listato righe 820-840.
Ciò che in pratica si ottiene è l'equivalente di un file VARIABLE, con la stessa caratteristica di compattamento. Al momento della OLD, si ricerca innanzi tutto il primo spazio vuoto (ASCII 32). Al byte precedente termina la lunghezza della stringa che segue. Si calcola la lunghezza della stringa [LU = VAL (LU$0)], e si ottiene la restituzione della stringa originaria [Q$(Z) = SEG$(G$,LU-LU + 2,LU)]. La parte letta viene tolta dal record. [G$ = SEG$(G$,LU-LU + 2 + LU,LEN(G$)] e si continua come sopra fino a che il record è finito [SEG$(G$,1,1) = " "OR SEG$(G$,1,1) = " "]. Vedi listato righe 960-1000.
Il compattamento descritto offre il vantaggio di poter processare mediamente 12-13 righe con una sola PRINT o INPUT. Ciò, in cambio, riduce di 10 volte il tempo necessario per le operazioni di file su cassetta, che, come noto, sono lunghe e tediose, e impiegano la massima parte del tempo perché la testina del registratore arrivi alla posizione del successivo record del file sul nastro.
La routine presentata è utile se le stringhe passate su nastro sono tutte molto corte (istruzioni ASSEMBLER sono mediamente di 8-15 caratteri). Sarebbe inutile se le stringhe da salvare fossero tutte, ad esempio, di 80 caratteri, come nel caso di un normale WORD-PROCESSOR: in questo caso la routine descritta non farebbe risparmiare alcun tempo.

USO DI XEDITOR

Dare il RUN. Si presenta lo schermo principale e appaiono dei simboli:
1) LC: indicherà il numero di linea del file che è attualmente sotto cursore.
2) UL: indicherà l'ultimo numero di linea del file.
Apparirà una domanda: File Nuovo o Vecchio? (N/V) V. In caso di risposta «N» si passa direttamente all'EDITOR vero e proprio. Con una risposta «V», invece, viene chiesto se l'INPUT del Vecchio file deve avvenire da disco o da cassetta: OLD DISK o CASS.? (D/C) C.
Scegliere secondo le proprie esigenze. Il cursore si posizionerà quindi su una riga completamente vuota. Inserire qui CS1 oppure DSK1. NOMEFILE, ove NOMEFILE va cambiato con il nome effettivo del file che si intende usare e premere ENTER. Il file verrà caricato in memoria.
In ogni caso, sia che abbiate scelto Vecchio file o Nuovo file vi troverete poi al quadro EDITOR vero e proprio.
Il cursore sarà posizionato al centro dello schermo (dodicesima riga). Avete a disposizione 28 caratteri per riga. Tutte le funzioni normali dell'edit sono abilitate. Battete la vostra sorgente, riga dopo riga.
Queste sono le funzioni dei tasti:1) Per passare alla riga successiva si preme ENTER. Il cursore rimane sempre sulla stessa linea (la dodicesima) e tutto lo schermo effettua lo scroll verso l'alto.
2) per ritornare sulla riga precedente tenere premuto FCTN E (freccia in su).
3) Per abilitare le funzioni speciali tenere premuto FCTN X (freccia in giù).

FUNZIONI SPECIALI (SPECIALS) DI XEDITOR

Scegliendo le funzioni speciali si apre una finestra in alto sul video, di tre righe. Appare così:
U(p or D(own, N(line, L(ine
R(eplace, M(ove, I(nsert
S(ave, O(ld, E(rase, F(ine
Impostando la lettera maiuscola che precede il nome di una funzione e premendo ENTER si attua la scelta della stessa funzione. Premendo FCTN 3 (ERASE) e poi ENTER si ha il ritorno al quadro precedente. Quest'ultima regola vale anche per tutte le sotto-funzioni.
Queste sono le spiegazioni delle funzioni speciali:
Up: effettua lo scroll verso il basso di 23 linee. Si ritorna così alle 23 linee precedenti.
Down: effettua lo scroll di 23 linee verso l'alto. Si raggiungono così le 23 linee successive.
Nline: mostra il numero di linea delle linee presenti sullo schermo, se esse non erano visualizzate, ma le cancella dallo schermo se esse erano visualizzate. In pratica è un interruttore a «TOGGLE», accende se spento, spegne se acceso. I numeri di linea appariranno a fianco di ciascuna riga, nella colonna 2. Sono così indicate solo le unità del numero di linea. Il valore delle decine ed eventuali centinaia va desunto dal numero della linea cursore (LC), che è sempre presente in alto a destra sullo schermo.
Se Nline viene mantenuto «acceso» si allunga, ovviamente, il tempo necessario per lo scroll.
Line: Sposta del numero di linee richiesto, verso le linee successive se si fornisce un valore positivo, verso le precedenti se si fornisce un numero di linee negativo. Ex. + 7 fa apparire sotto il cursore la settima riga dopo quella che al momento è sotto il cursore. Oppure, − 7 fa apparire la settima riga che precede quella che al momento è sotto il cursore.
Replace. Rimpiazza una stringa (o segmento di stringa) dalla prima linea fino all'ultima. Apparirà: Replace string /OLD/NEW/. Mettere al posto di OLD la stringa che si vuol mutare e al posto di NEW la stringa che si vuol immettere al posto della vecchia e premere ENTER. Apparirà: /OLD/NEW/
Replace [S/N/A(11/B(asta]
Inoltre, un carattere a forma di cursore si posizionerà a sinistra dell'inizio della stringa OLD in una linea del testo. Ora:
1) S (per Si) farà cambiare la OLD con la NEW, e i caratteri eventualmente eccedenti il 28esimo, nella nuova linea di testo così formata, saranno eliminati.
2) N (per No) farà passare all'eventuale successiva.
3) A (per All = TUTTI) innesca un procedimento automatico, per cui tutte le stringhe uguali alla OLD saranno cambiate nella NEW, dalla prima all'ultima linea di testo.
4) B (per Basta) determina l'uscita dall'opzione.
Nota che solo la prima occorrenza della OLD in ogni linea viene proposta per lo scambio con la NEW. Se si vogliono cambiare anche le successive occorrenze bisogna ripetere il procedimento.
È da rimarcare che la REPLACE può essere vantaggiosamente impiegata per ricercare una determinata stringa in tutto il testo. Allo scopo, immettere sia come OLD che come NEW la stessa stringa, per sicurezza, e poi iniziare la ricerca, premere N fino a che non si è trovata la linea di testo voluta. Guardare in che zona si trova, e poi premere B (per Basta).
Move: Muove una linea, o un determinato numero di linee, da una zona ad un'altra del testo. Ex.: impostando 4,7,2 e premendo ENTER si ottiene l'in-

serimento delle linee 4-5-6-7 tra la linea 2 e la linea 3.
Se si vuol spostare solo una linea, lo stesso numero va ripetuto. Ex. 13,13,17 sposta la linea 13 e la mette dopo la linea 17.
Nota: perché il formato dell'istruzione sia valido occorre sempre fornire tre valori numerici, separati tra di loro da due virgole. La stessa sintassi sarà poi necessaria anche per la «OLD» e la «SAVE» (vedi).
Insert: inserisce nuove linee vuote sotto la LINEA CURSORE. Il valore da fornire è il numero di nuove linee desiderate.
Save: permette il SAVE del file su disco (D), cassetta (C), o su carta (S) [cioè Stampa]. È possibile un SAVE totale (ex. CS1) oppure parziale, ex. 34, 45, CS1. In quest'ultimo caso solo le linee comprese tra la linea 34 e la linea 45 saranno salvate, incluse le due estreme (ovvero la 34 esima e la 45 esima).
Se si sceglie l'opzione (S), stampa su carta, si può anche far apparire il numero di linea a fianco a ciascuna linea di testo (appare un'apposita opzione).
Old: si può scegliere fra disco (D) o cassetta (C): appare una linea vuota, nella quale potete inserire il nome del vostro file. Avete queste ulteriori opzioni:
1) CS1 oppure DSK?. NOMEFILE: carica il file indicato, che annulla quello precedentemente in memoria.
2) 34, CS1 oppure 34, DSK?. NOMEFILE: effettua il MERGE TOTALE fra il file in memoria e quello da caricare. Il file da caricare andrà a inserirsi tra la riga 34 e la seguente (riga 35).
3) 17,52,CS1 oppure 17,52,DSK?. NOMEFILE: carica il file indicato, a partire dalla linea 17 fino alla linea 52 inclusa. Il file presente in memoria viene annullato:
4) 21,3,57,CS1 oppure 21,3,57,DSK?. NOMEFILE: carica il file indicato a partire dalla linea 3 fino alla linea 57 inclusa e lo immette nel file presente in memoria, inserendolo tra la linea 21 e la seguente (linea 22). Quest'ultimo formato permette cioè un MERGE PARZIALE (selettivo).
Potete variare i numeri forniti negli esempi, ovviamente!
Erase: cancella una o più linee, come indicato dal valore che fornirete, a partire dalla linea cursore. Il file in memoria viene ricompattato. Nota: la stessa funzione non può essere ottenuta con una ripetuta azione del tasto FCTN 3, perché questo lascerebbe linee vuote dietro di sé nel file.
Fine: si può scegliere in questa sede se si vuole effettivamente finire, con il che si arriva alla END del programma, o si vuole un Nuovo testo, con il che il file presente in memoria viene cancellato, e si può ricominciare con un nuovo testo. Il file precedente viene irrimediabilmente perso in entrambi i casi. Effettuate una SAVE in precedenza, se ciò che avete battuto sulla tastiera valeva qualcosa. Alternativamente, invece delle due precedenti opzioni, si può continuare con lo stesso testo.

QUALCHE SUGGERIMENTO

A) Il nome della stampante impiegata (valore DEFAULT) si trova a riga 170, ed è attualmente sistemata a «RS232. BA=4800. DA=8». Ognuno varia questo valore in accordo con la propria (eventuale) stampante.

B) Non è presente nel listato una istruzione ON BREAK NEXT, anche se sarebbe utile. La dimenticanza è solo apparente. È bene infatti che voi immettiate questa istruzione dopo che sarete certi di aver ottenuto una copia che «gira» perfettamente. Altrimenti al primo «RUN» potreste cascare in un ciclo senza uscita che vi obbligherebbe a spegnere il computer e, addio, lavoro eseguito...
La ON BREAK NEXT può essere convenientemente immessa a linea 160 del listato.

C) Io, al vostro posto, non batterei all'inizio nessuna delle ON ERROR presenti nel listato. Esse si trovano alle seguenti linee: 160, 1290, 1300, 1370, 1470. Se le batterete, non capirete mai dove avete commesso eventuali errori di battitura e la ricerca degli errori è in queste condizioni perlomeno proibitiva. Solo quando sarete sicuri che tutto è OK, vi conviene immettere le ON ERROR ai numeri di linea suriportati (vedi il listato per la posizione esatta).

D) XEDITOR funziona anche senza espansione di memoria. Chi però ha sempre l'espansione di memoria inserita può aggiungere al listato la classica inibizione del FCTN QUIT. Un FCTN QUIT accidentale dopo ore di lavoro è una vera dannazione. Per coloro che ancora non conoscono come inibire il FCTN QUIT riportiamo la sequenza:
CALL INIT:: CALL LOAD (-31806,16)
Potrete inserire questa sequenza a linea 160.

SCHEMA DEL LISTATO A BLOCCHI

100- 140 Intestazione
150- 200 Inizializzazione
210- 330 Routine principale di XEDITOR
340- 380 Selezione Opzioni (SPECIALS) e scroll variabile
390- 420 Inserimento nuove linee
430- 460 Cancellazione linee
470- 480 Scroll DOWN di 23 linee
490- 510 Scroll UP di 23 linee e Scroll variabile
520- 540 Mostra numeri di linea
550- 650 Replace String
660- 760 Muove linee
770- 940 SAVE ROUTINE
 810- 840 Save su cassetta
 870- 910 Save su disco o su carta
 920- 940 Selezione Stampante
950-1180 OLD ROUTINE
 950-1000 Old cassette
 1010-1090 Selezione tipo di Old
 1100-1140 Old da disco
 1150-1180 Risistema file in memoria dopo il MERGE
1190-1220 Limiti display at
1230-1240 Visualizza numeri linee e cursore fermo
1250-1270 Aspetta che tu prema un tasto
1280-1300 Routine di errori
1310-1350 Routine di FINE
1360-1470 Smistamento istruzioni con virgole
1480 Fine programma

TI99/4A

ORGANIZZAZIONE DELLA MEMORIA DEL TI-99/4A

Alcuni lettori chiedono ragguagli circa l'organizzazione della memoria del TI-99/4A. Forniamo le seguenti sommarie informazioni, desunte, perlopiù, dalla MILLER GRAPHICS, USA. e, poche per la verità, dalla personale esperienza del sottoscritto.
Il microprocessore TMS9900 è capace di indirizzare direttamente 64 kbytes di memoria (CPU RAM). Inoltre, nel TI-99/4A, sono state previste delle memorie «esterne», indirizzabili indirettamente a mezzo di «porte di accesso». Queste memorie esterne sono:
1) VDP RAM, size 16 kbytes. Indirizzi utili: minimo >0000, massimo >3FFF. È gestita dal TMS9918 in base ai comandi che riceve dal TMS9900. Le «porte di accesso» per la VDP si trovano agli indirizzi >9C02, >9C00, >8800 CPU RAM.
2) GROM, size 18 kbytes nella consolle. Inoltre, nei Moduli (SSS) possono trovare posto ulteriori GROM, per altri 30 kbytes. Il size totale delle memorie GROM ammonta così a 48 kbytes. Indirizzi utili:

GROM 0 da >0000 a >17FF:
GROM 1 da >2000 a >37FF:→ GROM DELLA CONSOLLE
GROM 2 da >4000 a >57FF:

GROM 3 da >6000 a >77FF:
GROM 5 da >8000 a >97FF:
GROM 6 da >A000 a >B7FF:→ GROM DEI MODULI
GROM 7 da >C000 a >D7FF:
GROM 8 da >E000 a >F7FF:

3) SPEECH SINTETIZER: size 32 kbytes. Indirizzi utili: Minimo >0000, massimo >7FFF.

Forniamo ora una mappa molto sommaria dei 64 k indirizzabili direttamente dal TMS9900 nel TI-99/4A (CPU RAM).

>0000->1FFF: 8 K ROM	Interprete GPL, interprete BASIC (parte).
>2000->3FFF: 8 K RAM	Abilitata se si possiede la Expansion Memory da 32 K. È la LOW-MEMORY (memoria BASSA).
>4000->5FFF: 8 K ROM	L'allacciamento con questa zona di memoria avviene a mezzo di CRU (Communication Register Unit). Sono banchi di memoria che risiedono nelle CARD da inserirsi nel PERIPHERAL EXPANSION BOX, quali il DISK CONTROLLER CARD, la RS232 CARD.
>6000->7FFF: 8 K ROM	È la classica zona delle ROM dei moduli (e della piccola RAM della MINI-MEMORY). Alcuni moduli hanno «impaginato» in questa zona più banchi di memoria, per un totale di 12 K, vedi il Modulo Ext. BASIC, o di 16 K, vedi alcuni moduli ATARI.
>8000->9FFF: 100 bytes di RAM!	Esatto, solo 100 bytes di RAM. È il famoso SCRATCH-PAD, l'unica e sola area di lavoro, multiuso, che tutti i moduli senza espansione possono usare. Credetelo pure, per questi ultimi non ci sono altre zone RAM «DIRETTAMENTE» impiegabili. Per ogni ulteriore fabbisogno di RAM essi si devono servire della VDP RAM (memoria «INDIRETTA»), così come fa il TI BASIC in qualunque configurazione esso venga impiegato. L'indirizzo della SCRATCH-PAD si trova tra >8300 e >83FF. Tutto il resto dell'area tra >8000 e >9FFF è pressoché vuoto. Contiene solo le «PORTE DI ACCESSO» a: 1) VDP RAM 2) GROM 3) SPEECH SINTETIZER
>A000->FFFF: 24 kbytes	Abilitata se si possiede la Exp. Memory da 32 K. È la HIGH MEMORY (memoria ALTA).

MODULI E GROM: UNA LOTTA ALL'ULTIMA ASTUZIA
Tutti i Moduli SSS marcati TEXAS devono avere perlomeno una GROM. Moduli con una sola GROM sono: MINI-MEMORY, EDITOR/ASSEMBLER, TI-WRITER, DISK-MANAGER, MULTIPLAN, etc... Alcuni moduli hanno quattro GROM (vedi Modulo Extended BASIC) e altri ancora hanno tutte le 5 possibili GROM (vedi EARLY READING).
Generalmente, i moduli delle case indipendenti non hanno invece GROM (vedi ATARI). Questo fatto ha determinato la produzione della consolle di tipo 1983 V2.2, rilasciata dalla TEXAS INSTR. Poco prima di uscire definitivamente dal mercato del TI-99/4A, con il preciso scopo di far piazza pulita dei concorrenti. La consolle 1983 V2.2 è capace di rifiutare i moduli che non contengono GROM, quindi, in pratica, di impedire l'uso dei moduli ATARI e ATARI simili. Per superare questo nuovo ostacolo introdotto dalla Texas Instr., sono stati escogitate diverse soluzioni, quali il «GROMBUSTER» della NAVARONE INDUSTRIES, USA. Niente a che vedere con «GOSTBUSTERS», anche se si può obiettare che i «fantasmi» delle GROM ci disturbano ancora. Il GROMBUSTER, attaccato sul fianco del computer, permette l'utilizzo dei moduli privi di GROM anche con le consolle 1983 V2.2.
Peccato che la Texas abbia cessato la produzione del TI-99/4A. Ero curioso di vedere quale sarebbe stata la mossa successiva della Texas in questa fine lotta all'ultima protezione.

PICCOLA MAPPA DI MEMORIA, MODULO EXT.
BASIC INSERITO + EXP. MEM. 32K

SEZIONE CPU RAM

Locaz.	Contenuto	
>2000	>205A	Puntatore alla CALL LINK-DEF NOME. È sempre usato per l'ingresso diretto in un programma in L.M. da CSALOCO, e poi ripristinato al suo normale valore (>205A è, in bytes decimali 32,90). >2000 è 8192 dec.
>2002	>24F4	Inizio di caricamento (DEFAULT) di un programma in L.M. generato da XASSEMBLE o di qualsiasi file assembler DIS/FIX 80, privo di AORG, caricato da dischetto subito dopo la CALL INIT. Dopo il caricamento la locazione a >2002 conterrà l'indirizzo della prima locazione LIBERA per un successivo caricamento di un eventuale ulteriore programma in L.M. Ex: se il primo programma era >102 bytes, dopo il caricamento in >2002 troverete >25F6.
>2004	>4000	Questa locazione contiene >4000 subito dop il CALL INIT, poi punterà all'ULTIMO WORD LIBERO in BASSA MEMORIA, prima dell'inizio della DEF TABLE.
>2006	>AA55	IDENTIFICATORE PER L'ABILITAZIONE DI PROGRAMMI IN L.M.: se diverso da >AA55 non saranno più abilitate le CALL LOAD e le CALL LINK (occorrerebbe re-ini-

	ziare la memoria bassa con una CALL INIT, con il ché tutto il contenuto della memoria da >2000 fino a >26FF sarebbe perso). Non modificare quindi il valore >AA55 per nessun motivo.
>2008 >23FF	UTILITIES VMBW, VSBW, VWTR, VMBR, VSBR, KSCAN, STRREF, NUMREF, STRASG, NUMASG. Nota: mancano in EXT. BASIC tre routines in MACRO-ASSEMBLER: GPLLNK, DSRLNK, LOADER.
>24FO >0000	Dovrà contenere l'indirizzo di inizio di programma per un «CSAVE» valido in XASSEMBLE. Questa locazione è ignorata dalle routines TEXAS.
>24F2 >0000	Dovrà contenere l'indirizzo di fine programma per un «CSAVE»» valido in XASSEMBLE. Anche questa locazione è ignorata dalle routines TEXAS.
>24F4 >3FF7	>24F4->3FF8 è la massima estensione di un programma valido da generarsi con XASSEMBLE. Qui risiederà il programma che creerete. La prima locazione utile, per un programma senza AORG sarà quindi >24F4, l'ultima >3FF7.
>3FF8 >3FFF	>3FF8 - >3FFF è la minima estensione della DEF TABLE, per un valido CSAVE con XASSEMBLE. Ovvero, è lo spazio occupato da una sola DEF. Ogni DEF aggiunta occuperà altri 8 bytes verso il basso, e sposterà quindi verso il basso di 8 bytes il puntatore all'ULTIMA MEMORIA LIBERA, che si trova a >2004. Degli 8 bytes occupati da ogni DEF, 6 bytes sono utilizzati per il nome della DEF, gli altri due bytes sono utilizzati per contenere l'indirizzo a cui si riferisce la DEF in questione (punto di ingresso al programma).

SEZIONE VDP RAM

>0000->02FF	ZONA SCHERMO per Ext. BASIC, Ed/Assembler, Minimemory. Sono 768 bytes. Se il VDP WRITE ONLY REGISTER 2 è diverso da zero, la ZONA SCHERMO non è più tra >0000 e >02FF (vedi manuale Ed.Ass. pag. 327).
>0300->03F7	SPRITE ATTRIBUTE LIST per Ext. BASIC, Ed/Assembler, se il VDP WRITE ONLY REGISTER 5 contiene >06 (vedi Man. Ed/Ass. pag. 327). Contiene: posizione SPRITE (riga, colonna), colore dello SPRITE, e, infine l'EARLY CLOCK, che determina la posizione dello SPRITE al fine della COINCIDENZA fra due SPRITES diversi.
>0380->077F	Valori interprete BASIC, PATTERN DESCRIPTOR TABLE e, anche, SPRITE DESCRIPTOR TABLE (zona DEFINIZIONE CARATTERI) per l'Ext. BASIC (se il VDP WRITE ONLY REGISTER 4 contiene >00 e il VDP WRITE ONLY REGISTER 6 contiene >00). In Assembler, invece, questa zona è esclusivamente dedicata alla SPRITE DESCRIPTOR TABLE (se il VDP WRITE ONLY REGISTER 6 contiene >00).

Il TMS 9918 (Video Processore Grafico del TI-99/4A), deputato a tutti i mutamenti che avvengono in VDP RAM, visualizza un'immagine sul video nella seguente maniera: va a vedere l'ASCII contenuto in ognuna delle 768 locazioni della ZONA SCHERMO (tra >0000 e >02FF per il valore DEFAULT). Per ogni valore trovato va a cercare la definizione corrispondente (8 bytes) nella PATTERN DESCRIPTOR TABLE (in questa tavola).

Quindi, fa apparire sullo schermo video, nella colonna e nella riga indicata, la definizione trovata nella PATTERN DESCRIPTOR TABLE.

Ex. La locaz. VDP RAM >0002 contiene >80 (dec. 128). Gli 8 bytes interessati alla definizione saranno a: >80*8+BASE. Ora, BASE è >0000 in Ext. BASIC (valore VDP WRITE ONLY REGISTER 4 moltiplicato per >800). Quindi, nell'esempio, si ha che >80*8+>0000=>80*8=>400 Ciò significa che il TMS9918 userà per la riga 1, colonna 3 (locaz. VDP >0002), la definizione contenuta tra >400 e >407 VDP RAM. Se questi bytes sono tutti >00 avremo la visualizzazione di un BLANK (SPAZIO) a riga 1, colonna 3 dello schermo. Se invece, i bytes tra >0400 e >0407 VDP RAM sono diversi da zero, avremo la visualizzazione, sempre a riga 1, colonna 3, di ciò che risulterà dalla nuova definizione.

Valgono, allo scopo di ridefinire i caratteri, le definizioni usate a mezzo di CALL CHAR (A$) riportate a pag. 83 e segg. del manuale del TI-BASIC (Manuale di color verde pisello, in italiano, fornito insieme alla consolle dalla Texas Instr.).

Per esempio, mettere a >400 VDP questa successione di caratteri >FF, >00, >00, >00, >00, >00, >00, >00, farà apparire una riga al posto di ogni spazio (ASCII >80, dec. 128) che sarà trovato tra >0000 e >02FF VDP RAM.

In Ext. BASIC gli ASCII dei caratteri in ZONA SCHERMO sono tutti aumentati di >60 (dec. 96). Questo permette lo spostamento dell'effettivo inizio della PATTERN DESCRIPTOR TABLE da >0000 a >0400 VDP RAM e oltre. Ecco perché lo SPAZIO, (valore ASCII normale >20 (dec. 32) ha in Extended BASIC il valore di >80 (dec. 128), che è dato da >20+>60 di OFFSET.

Se i valori ASCII non fossero aumentati dell'OFFSET di >60, si avrebbe un conflitto tra la ZONA SCHERMO e la PATTERN DESCRIPTOR TABLE. Infatti la stessa zona di memoria VDP RAM (>0000->02FF dovrebbe contenere, allo stesso tempo, sia l'ASCII che la sua stessa definizione. Pensate ad un valore ASCII di >21 che si trovi a locazione >108 VDP RAM (riga 9, colonna 8): la sua definizione dovrebbe iniziare a >21*8=>108 VDP RAM, ov-

TI 99/4A

vero nella stessa locazione che contiene il valore ASCII. La cosa è ovviamente impossibile. La Texas Instr. ha spostato la PATTERN DESCRIPTOR TABLE sopra i >400 VDP RAM, in ambiente Ext. BASIC, per risparmiare memoria VDP, che, non dimentichiamo, è l'unica atta a contenere il programma Ext. BASIC se non si possiede l'espansione di memoria.

>0780->07FF	SPRITE MOTION TABLE. Questa zona è deputata al controllo delle velocità degli SPRITES. IL TMS 9918 (VIDEO PROCESSORE GRAFICO) non può utilizzare un'altra zona per la velocità degli SPRITES. Questo è il motivo per cui, a mio parere, non si possono avere SPRITES con movimento automatico in BIT MAP MODE.
>0800->081F	COLOR TABLE in Extended BASIC, se il VDP WRITE ONLY REGISTER 3 contiene >20. In Assembler la COLOR TABLE si trova a >0380, in quanto il VDP WRITE ONLY REGISTER 3 contiene >0E.
>0820->08BE	CRUNCH BUFFER. In Ext. BASIC questa zona è usata per generare i TOKEN del BASIC partendo dalle istruzioni delle linee di un programma BASIC al momento dell'immissione delle stesse da tastiera.
>08C0->095B	REDO BUFFER. Quando in Ext. BASIC, in modo COMANDO, premete REDO (FCTN 8), riappare l'ultima istruzione immessa con ENTER. È da questa zona che viene prelevata la vecchia istruzione immessa. I valori ASCII di ogni carattere presente in questa zona sono aumentati di >60. Nota che gli indirizzi di inizio e fine schermo, su cui deve riapparire la stringa si trovano rispettivamente a >038C e >038E VDP RAM.
>095C->37D7	ZONA PER STRINGHE E ARRAYS NUMERICI.
>37D8->3FFF	ZONA DISK CONTROLLER PER CALL FILES (1).

VSBW, VMBW, VSBR, VMBR e OFFSET in Ext. BASIC.

In Est. BASIC, la zona definizioni caratteri (PATTERN DESCRIPTOR TABLE), come detto, inizia effettivamente a >0400 VDP RAM (ASCII 32). Le routines MACRO ASSEMBLER VSBW, VMBW NON aggiungono l'OFFSET necessario al valore ASCII da immettere in VDP RAM.

Per esempio, la seguente routine

```
TESTO   TEXT    «ABAB»
INIZ    LI      RO, >0002
        LI      RI, TESTO
        LI      R2,4
        BLWP    @VMBW
```

non farà apparire la scritta «ABAB» (in alto a sinistra sullo schermo). Per ottenere una visualizzazione effettiva conviene non usare la VMBW, ma la VSBW, in questa maniera:

```
TESTO   TEXT    «ABAB»
INIZ    LI      RO, >0002
        LI      R3, TESTO
        LI      R2,4
        LOOP    MOVB*R3+,R1
        AI      RI,>6000
        BLWP    @VSBW
DEC     R2
        JNE     LOOP
```

Analogamente, non ci si può aspettare che una VMBR restituisca valori ASCII normali. Infatti, essi saranno passati in CPU RAM così come essi si trovano in VDP RAM, e cioè aumentati di >60. Una routine di questo genere permetterà di superare l'ostacolo.

```
BUFFER  BSS     4
INIZ    LI      RO,>0002
        LI      R3, BUFFER
        LI      R2,4
LOOP    BLWP    @VSBR
AI RI,>A000 * decrementa di >60 il Most significant
MOVB RI,*R3+ byte di R1
DEC R2
JNE LOOP
```

Si possono costruire routines molto più veloci, ma gli esempi forniti sono sufficienti per comprendere il concetto dell'OFFSET per i valori ASCII in VDP RAM con l'Ext. BASIC.

di LETIZIA BIZZARRI

SHARP

PARLIAMO DI SHARP

Anche se con un mese di ritardo, eccoci di nuovo qui. Sembra che la rubrica stia riscuotendo un discreto successo, ma ho l'impressione che non sia ancora conosciuta dai possessori di computer Sharp: spero che la voce si sparga, in modo da sentirvi sempre più entusiasti e numerosi rispondere al mio appello, che vi ripeto qui: se avete qualche problema con il vostro Sharp, se avete scoperto qualche trucchetto nuovo, se c'è un programma che non vi gira, scrivete pure: sono qui proprio per rispondervi, e queste pagine diventeranno molto più vive e attuali con la vostra collaborazione. Come era prevedibile la parte del leone l'ha fatta l'MZ700: gli utenti di MZ80K/A/B mi vorranno perdonare se fin da questo numero sempre maggior spazio sarà dedicato a questo computerino.

Due sono le cose importanti delle quali volevo parlarvi prima di iniziare a trattare argomenti più strettamente tecnici: 1) la creazione di un club di utenti Sharp per scambiare programmi; 2) il nuovo computer Sharp della serie MZ800.

Parliamo di club: la lamentela più frequente che riecheggia fra gli sharpisti è la difficoltà a reperire del software.

Dalla prossima volta sarò in grado di presentarvi e recensire il software com-